REMARKS

Claims 1-22 are now in the application. Claims 1, 15, 16 and 17 have been amended. Applicants respectfully request reconsideration of the application, as amended, in view of the following remarks.

Applicants wish to thank Examiner Kiss for the telephonic interview on July 12, 2006.

The claims were rejected under 35 U.S.C. §101. This rejection was discussed in the telephonic interview. The claims have been amended to include language suggested in the office action. Among other things, the claims now recite that the methods or utilities generate a report that is viewable by a developer-user, so the developer-user may address the vulnerabilities identified in the report by modifying the source code listing if necessary. The present application discusses this in paragraphs 89 and 93, among other places, and the entire application is clearly directed to a development tool to analyze source code so a user-developer may address potential vulnerabilities detected by the utility. The provisional application includes many pages directed to the utilities user interface and assessment reports, as well, see, e.g., pages 3, 7, 30-31 of the provisional specification.

The claims were rejected under 35 U.S.C. §112, second paragraph. The claims have been amended accordingly.

Claims 1-16 were rejected under 35 U.S.C. §102(b) as being anticipated by David Wagner, et al. This rejection was discussed in the telephonic interview.

In short, as discussed at the personal interview of January 19th, the prior amendment, and the telephonic interview of July 12, Wagner et al. uses constraint language and formal methods to detect the existence of buffer overflow vulnerabilities in C source code. Wagner et al.

converts C source code into a constraint language, and analyzes the constraint language to determine the existence of buffer overflow vulnerabilities.

In contrast, the invention is directed to a method that uses compiler-like techniques to detect vulnerabilities. Applicants attempted to capture that distinction in the prior amendments by specifically referencing analysis in the context of the source code's control and data flow. In the telephonic interview, applicants proposed making this distinction clearer by amendment, which the current claims attempt to do. The present claims recite specifically the analysis of the source code listing's control and data flow to analyze the source code listing or the variables in the source code listing. Analyzing control flow and data flow are specific compiler-like techniques that attempt to understand the semantics of the source code listing. In compiler technology this analysis is done for code generation. In the invention, this is done to detect vulnerabilities in the source code (not for code generation). This aspect is discussed throughout the specification, see, e.g., paragraphs 40, 112, and 114.

Claims 17-19 were rejected under 35 U.S.C 103(a) as being unpatentable over Wagner in view of Viega. More specifically, the office action relied on Wagner for all elements of claim 17 except for the database of vulnerabilities which the action cited Viega as disclosing.

As explained above, Wagner does not teach or suggest analyzing the control flow and data flow of a source listing to detect vulnerabilities. Therefore, the amended claims should be allowed for this reason alone. In addition, Viega explicitly teaches away from using real compiler techniques to check for vulnerabilities in the source code. Viega explains:

> "One reason we chose *not* to use a "real parser" was because we wanted to have a false negative rate as close to 0 as possible. Analysis tools using traditional parsing (such as the `lint` family of tools) can only analyze a single build of a program at once...we also want to examine the entire program easily without having to

> specify multiple build configurations and keep track of uncovered code" (page 3, emphasis added).

To minimize the false negative rate, Viega rejects the use of real compiler techniques. Viega highlights the disadvantages of analysis tools using traditional parsing. Further rejecting real compiler techniques in source code vulnerability scans, Viega explains:

> "Another reason for not using 'real' context-free parsing is that that we wanted to be able to support interactive programming environments such as Emacs and Microsoft Visual C++ in real time...as the programmer enters code, the programming environment should recognize the likelihood that a particular piece of code contains a security problem and act appropriately. Unfortunately, traditional parsing techniques are not suitable for meeting this goal because they only work reliably on semantically valid programs. Furthermore, highly accurate error handling in traditional parsers is notoriously difficult" (page 3).

By highlighting the importance of meeting the needs of interactive programming environments and detailing the insufficiencies of traditional parsing techniques, Viega teaches away from the use of real compiler techniques. In light of the above passages, the applicants believe Viega does not supply that which is missing from Wagner and that claims 17-19 are allowable.

Claims 18-21 were rejected under 35 U.S.C. 103(a) as unpatentable over Wagner in view of LaRochelle et al. Although the office action cited claims 18-21, the substance of the office action and the claim language indicate that claims 20-22 were intended. Accordingly, the following response considers the rejection of claims 20-22 as unpatentable over Wagner in view of La Rochelle under 35 U.S.C. 103(a). More specifically, the office action relied on Wagner for all elements of claim 20-22 except for identifying the location of the vulnerability which the office action cited LaRochelle as disclosing.

As explained above, Wagner does not teach or suggest analyzing the control flow and data flow of a source code listing to detect vulnerabilities. Thus, the amended claims should be allowed for this reason alone. In addition, LaRochelle does not supply all that is missing from

Wagner. Instead of using source code to identify vulnerabilities, LaRochelle develops a means for manual intervention. The developer must enter specific comments to specify the constraints in checking vulnerabilities. These semantic comments are termed "annotations". LaRochelle discloses:

> "Our approach is to exploit semantic comments (henceforth called *annotations*) that are added to source code and standard libraries. Annotations describe programmer assumptions and intents" (page 3).

Therefore, in the method disclosed by LaRochelle, a programmer uses annotations to manually perform the same function automatically performed by the source code in the disclosed invention.

In view of the amendments and remarks made above, applicants believe the pending application is in condition for allowance. The Commissioner is hereby authorized to charge the required fee of $60.00 for filing the request for one-month extension of time to our Deposit Account No. 08-0219. Please apply any charges not covered, or any credits, to Deposit Account No. 08-0219.

Respectfully submitted,

Date: SEP 19 2006

Peter M. Dichiara
Reg. No. 38,005

Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, MA 02109
Telephone: (617) 526-6466
Facsimile: (617) 526-5000